

# A New Foundation of Attack Trees in Linear Logic

Harley Eades III

Computer Science  
Augusta University  
harley.eades@gmail.com

**Abstract.** In this paper I provide an update on an ongoing research project investigating founding attack trees in the resource conscious theory called linear logic. I introduce a new linear logic called the Attack Tree Linear Logic (ATLL) which models attack trees as formulas and uses implication to reason about attack trees. I then introduce a new generalization of Bucciarelli and Ehrhards indexed linear logic to obtain Indexed ATLL which adds costs to attack trees. The most interesting aspect of Indexed ATLL is that implication is endowed with a relation for taking into consideration the costs while reasoning about attack trees.

## 1 Introduction

What is a mathematical model of attack trees? There have been numerous proposed answers to this question. Some examples are propositional logic, multisets, directed acyclic graphs, source sink graphs (or parallel-series pomsets), Petri nets, and Markov processes. Is there a unifying foundation in common to each of these proposed models? Furthermore, can this unifying foundation be used to further the field of attack trees and build new tools for conducting threat analysis?

The answer to the first question is positive, but the answer to the second question is open. Each of the proposed models listed above have something in common. They can all be modeled in some form of a symmetric monoidal category [10,1,3,4]. That is all well and good, but what can we gain from monoidal categories?

Monoidal categories are a mathematical model of linear logic as observed through the beautiful Curry-Howard-Lambek correspondence [8]. In linear logic every hypothesis must be used exactly once, and hence, if we view a hypothesis as a resource, then this property can be stated as every resource must be consumed. This linearity property is achieved by restricting the logic so that hypothesis are never duplicated or removed.

Multisets and Petri nets both capture the idea that the nodes of an attack tree consist of the attack action and the state – the resource – of the system being analyzed. As it turns out, linear logic has been shown to be a logical foundation for multisets [10] and Petri Nets [1]. Thus, linear logic has the ability to model the state as well as attack actions of the goals of an attack tree.

In this paper<sup>1</sup> I introduce a new logical foundation of attack trees – with and without costs – in linear logic; the type of attack trees considered in this paper are attack trees with sequential conjunction due to Jhawar et al. [6]. This new foundation is based on a new logic called the Attack Tree Linear Logic (ATLL).

In ATLL (Section 3) attack trees are modeled as linear formulas where base attacks are atomic formulas, and each branching node corresponds to a binary logical connective. Consider the attack tree for an ATM attack from Figure 1. We can model this attack tree as a formula in ATLL as follows:

$$\begin{aligned} B_1 &:= \text{“steal card”} \\ B_2 &:= \text{“social engineer”} \\ B_3 &:= \text{“find a post-it”} \\ B_4 &:= \text{“withdrawal money”} \\ T_1 &:= (B_1 \odot (B_2 \sqcup B_3)) \triangleright B_4 \end{aligned}$$

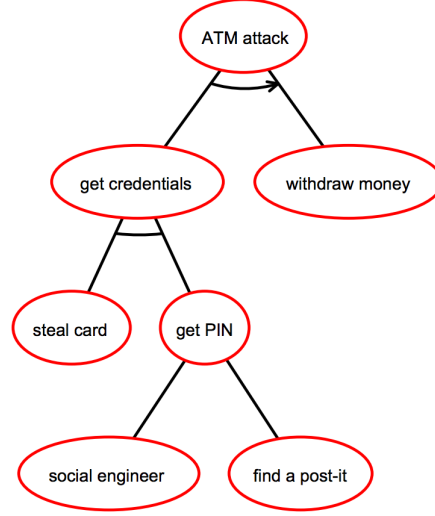
Each  $B_i$  is an atomic formula, parallel conjunction of attack trees is denoted by  $T_1 \odot T_2$ , choice between attacks by  $T_1 \sqcup T_2$ , and sequential conjunction of attacks by  $T_1 \triangleright T_2$ . Parallel conjunction and choice are both symmetric, but sequential conjunction is not. Now that we can model attack trees as formulas we should be able to use the logic to reason about attack trees.

Reasoning about attack trees corresponds to proving implications between attack trees. In fact, every equation from Jhawar et al.’s work on attack trees with sequential conjunction [6] can be proven as an implication in ATLL. Consider a second attack tree from Figure 2. We can model this attack tree as a formula in ATLL as well (the base attacks are the same):

$$T_2 := ((B_1 \odot B_2) \triangleright B_4) \sqcup ((B_1 \odot B_3) \triangleright B_4)$$

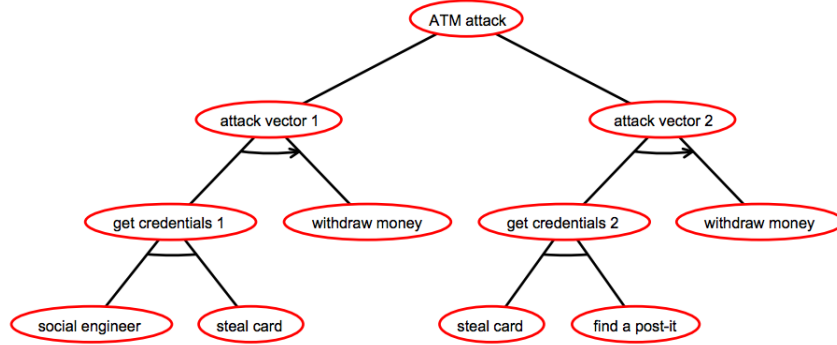
The attack trees  $T_1$  and  $T_2$  represents the same attack, and this can be proven in ATLL by showing that  $T_1 \multimap T_2$  where  $\multimap$  denotes bi-implication. The proof holds by using the distributive rules for choice.

In addition, implication can be used to prove subattack tree relationships between attack trees. For example, supposed we wanted to show that the tree  $B_1 \odot B_2$  is a subattack tree of  $T_2$ , then this is the case if  $T_2 \multimap (B_1 \odot B_2)$ .



**Fig. 1.** Attack Tree for an ATM attack from Figure 1 on page 2 of Kordy et al. [?]

<sup>1</sup> This material is based upon work supported by the National Science Foundation CRII CISE Research Initiation grant, “CRII:SHF: A New Foundation for Attack Trees Based on Monoidal Categories“, under Grant No. 1565557.



**Fig. 2.** Canonical Attack Tree for an ATM attack from Figure 2 on page 3 of Kordy et al. [?]

This requires some additional rules be added to ATLL and this is discussed in Section 3.

Using a generalization of Bucciarelli and Ehrhards [2] indexed linear logic I extend ATLL with costs. I call this extension Indexed ATLL (Section 4). In this system every base attack is annotated with a cost, and every branching node is associated with a binary operation used for computing the cost at that node. Then implication is associated with a binary relation on costs. This new binary relation is used to insure that the costs of two trees are related when proving implications between attack trees. Thus, one can prove facts about attack trees in Indexed ATLL that require one to reason about both the tree structure as well the quantitative data in the tree simultaneously.

Modeling attack trees in linear logic has a number of benefits. First, it connects attack trees back to logic. Kordy et al. [7] proposed that attack trees be modeled in propositional logic, but in this model attacks can be freely duplicated and contracted which goes against the process nature of an attack tree. However, linear logic restores this natural interpretation without losing the process interpretation of attack trees. By connecting attack trees to logic we can also tap into the long standing research and development of automation, for example, SAT, SMT, proof search, etc. Finally, by connecting to logic we also connect to the theory of statically typed functional programming through the Curry-Howard-Lambek correspondence which will allow for the development of a programming language that can be used to certify the correctness of attack trees and the analysis performed using them.

**Contributions.** This paper offers the following contributions:

- The Attack Tree Linear Logic (ATLL): a new linear logic for the specification and analysis of attack trees.

- The Indexed Attack Tree Linear Logic (Indexed ATLL): an extension of ATLL with costs and the ability to reason about them while reasoning about attack trees.

## 2 A Brief Introduction to Ordered Linear Logic

In order to aid the reader unfamiliar with linear logic and how it and other logics can be used as a modeling framework I first give a brief introduction to (ordered) linear logic and how it can be used to model various structures in computer science. I assume very little of the reader, and start with the basics of the specification of logics. Ordered linear logic (OLL) [9] will be the ongoing example throughout this section, because it is the logic ATLL is based on. If the reader is familiar with linear logic, then they can safely skip this section.

Every logic presented in this paper is in the form of sequent-style natural deduction. This type of formalization begins with the specifying the syntax of formulas and sequents. The latter is defined by a set of inference rules. The syntax for OLL is as follows:

$$\begin{array}{ll}
\text{(formulas)} & A, B, C ::= b \mid \top \mid A \odot B \mid A \triangleright B \mid A \times B \mid A \multimap B \\
\text{(ordered contexts)} & \Delta ::= \cdot \mid A \mid \Gamma, \Delta \\
\text{(unordered contexts)} & \Gamma ::= \cdot \mid A \mid \Theta, \Psi
\end{array}$$

Formulas of OLL consist of atomic formulas denoted by  $b$ , a symmetric tensor product  $A \odot B$ , its unit  $\top$ , a non-symmetric tensor product  $A \triangleright B$ , non-linear conjunction  $A \times B$ , and linear implication  $A \multimap B$ . Contexts are lists of hypothesis where  $\Delta$  is a list of ordered hypothesis, and  $\Gamma$  is a list of unordered hypothesis. The former are associated with the non-symmetric tensor product, and the latter with the symmetric tensor product. We denote the empty context by  $\cdot$ , and appending of contexts  $\Gamma_1$  and  $\Gamma_2$  by  $\Gamma_1, \Gamma_2$ .

First, some intuitions for the differences between each type of conjunction. One can view the two tensor products,  $A \odot B$  and  $A \triangleright B$ , as a form of linear conjunction. The former should be thought of as parallel conjunction, and not as the usual conjunction from classical logic. In fact, in linear logic it is not the case that  $(A \odot B) \multimap A$  nor  $(A \odot B) \multimap B$ . In addition, it is not the case that  $A \odot A$  is the same as  $A$ , because running a process in parallel with itself is not the same as running  $A$  once. The same can be said for  $A \triangleright B$ , but one should think of this operator as sequential conjunction, which means, that it is not the case that  $(A \triangleright B) \multimap (B \triangleright A)$ , and I prove this below. Now non-linear conjunction  $A \times B$  is ordinary conjunction from classical and intuitionistic logic.

Sequents of OLL are denoted by  $\Delta; \Gamma \vdash A$ . We say that the sequent  $\Delta; \Gamma \vdash A$  holds if and only if given the hypothesis in  $\Delta$  and  $\Gamma$  one can construct a proof of the formula  $A$  using the following inference rules:

$$\begin{array}{c}
\frac{}{;\cdot \vdash A} \text{id}_\odot \quad \frac{}{A; \cdot \vdash A} \text{id}_\triangleright \quad \frac{}{;\cdot \vdash \top} \top \quad \frac{\Delta_1; \Gamma \vdash A \quad \Delta_2; \Gamma \vdash B}{\Delta_1, \Delta_2; \Gamma \vdash A \times B} \times_I \\
\frac{\Delta; \Gamma \vdash A \times B}{\Delta; \Gamma \vdash A} \times_{E1} \quad \frac{\Delta; \Gamma \vdash A \times B}{\Delta; \Gamma \vdash A} \times_{E2} \quad \frac{\Delta_1; \Gamma_1 \vdash A \quad \Delta_2; \Gamma_2 \vdash B}{\Delta_1, \Delta_2; \Gamma_1, \Gamma_2 \vdash A \odot B} \odot_I \\
\frac{\Delta_1; \Gamma_2 \vdash A \odot B \quad \Delta_2; \Gamma_1, A, B, \Gamma_3 \vdash C}{\Delta_1, \Delta_2; \Gamma_1, \Gamma_2, \Gamma_3 \vdash C} \odot_E \quad \frac{\Delta_1; \Gamma_1 \vdash A \quad \Delta_2; \Gamma_2 \vdash B}{\Delta_1, \Delta_2; \Gamma_1, \Gamma_2 \vdash A \triangleright B} \triangleright_I \\
\frac{\Delta_2; \Gamma_1 \vdash A \triangleright B \quad \Delta_1, A, B, \Delta_3; \Gamma_2 \vdash C}{\Delta_1, \Delta_2, \Delta_3; \Gamma_1, \Gamma_2 \vdash C} \triangleright_E \quad \frac{\Delta; \Gamma_1, A, B, \Gamma_2 \vdash C}{\Delta; \Gamma_1, B, A, \Gamma_2 \vdash C} \text{sym}_\odot \\
\frac{\Delta; \Gamma, A \vdash B}{\Delta; \Gamma \vdash A \multimap B} \multimap_I \quad \frac{\Delta_1; \Gamma_1 \vdash A \multimap B \quad \Delta_2; \Gamma_2 \vdash A}{\Delta_1, \Delta_2; \Gamma_1, \Gamma_2 \vdash B} \multimap_E
\end{array}$$

An inference rule should be read from top to bottom as an implication. The sequents on top of the line are the premises, and the sequent below the line is the conclusion. For example, one should read the rule  $\odot_I$  as if  $\Delta_1; \Gamma_1 \vdash A$  and  $\Delta_2; \Gamma_2 \vdash B$  both hold, then  $\Delta_1, \Delta_2; \Gamma_1, \Gamma_2 \vdash A \odot B$  holds. An inference rule with no premises are called axioms. For example, the rules  $\text{id}_\odot$  and  $\text{id}_\triangleright$  are both axioms.

Inference rules are read from top to bottom, but applied from bottom to top. A sequent,  $\Delta; \Gamma \vdash A$ , holds if and only if there is a series of inference rules that can be composed into a derivation. Every derivation is a tree where the root is the sequent we wish to prove, and then rules are composed to form a tree. A rule can be composed with another if the formers conclusion matches a premise of the latter. If every branch of this tree reaches an axiom, then it is a valid proof, but if any branch gets stuck, then the sequent does not hold. The following is a proof that the symmetric tensor is indeed symmetric:

$$\frac{\frac{\frac{}{;\cdot \vdash B} \text{ID}_\odot \quad \frac{}{;\cdot \vdash A} \text{ID}_\odot}{;\cdot, A \vdash B \odot A} \odot_I}{;\cdot \vdash A \odot B \vdash A \odot B} \text{ID}_\odot \quad \frac{}{;\cdot, B \vdash B \odot A} \text{SYM}_\odot}{;\cdot \vdash A \odot B \vdash B \odot A} \odot_E \multimap_I$$

We call the root of the tree the goal of the proof, and as we construct a derivation this goal is refined into potentially several new subgoals. This is called goal directed proof. As we can see the previous derivation is a valid proof of the goal  $;\cdot \vdash (A \odot B) \multimap (B \odot A)$ .

The following derivation is an invalid proof that the non-symmetric tensor product is symmetric:

$$\frac{\frac{\frac{}{;\cdot \vdash A \triangleright B \vdash A \triangleright B} \text{ID}_\odot \quad A, B; \cdot \vdash B \triangleright A}{;\cdot \vdash A \triangleright B \vdash B \triangleright A} \triangleright_E}{;\cdot \vdash (A \triangleright B) \multimap (B \triangleright A)} \multimap_I$$

At this point we have refined our goal to the subgoal  $A, B; \cdot \vdash B \triangleright A$ , but this is impossible to prove, because the exchange rule  $\text{sym}_{\odot}$  cannot be applied to the ordered context to commute  $A$  and  $B$ .

Building proofs from the inference rules given above can often be tedious and long. To make this process easier it is often necessary to introduce new rules that can be proven to be valid in terms of the inference rules already given. Rules introduced in this way are called derivable inference rules. One derivable rule we will make use of is the following:

$$\frac{\Delta_2; \Gamma_2 \vdash A \multimap B \quad \Delta_1; \Gamma_1 \vdash B \multimap C}{\Delta_1, \Delta_2; \Gamma_1, \Gamma_2 \vdash A \multimap C} \text{comp}$$

Proving this rule is derivable amounts to creating a derivation whose branches terminate at either an axiom or the premises  $\Delta_2; \Gamma_2 \vdash A \multimap B$  or  $\Delta_1; \Gamma_1 \vdash B \multimap C$ . The proof is as follows:

$$\frac{\Delta_1; \Gamma_1 \vdash B \multimap C \quad \frac{\Delta_2; \Gamma_2 \vdash A \multimap B \quad \frac{}{\cdot; A \vdash A} \text{id}_{\odot}}{\Delta_2; \Gamma_2, A \vdash B} \multimap_E}{\Delta_1, \Delta_2; \Gamma_1, \Gamma_2, A \vdash C} \multimap_E \quad \frac{}{\Delta_1, \Delta_2; \Gamma_1, \Gamma_2 \vdash A \multimap C} \multimap_I$$

At this point I have introduced the basics of sequent-style natural deduction, but nothing I have said up to now has been specific to ordered linear logic. We call this logic ordered, because it contains the non-symmetric tensor product. Its non-symmetry is enforced by the separation of hypotheses. What makes a logic linear?

Linear logic was first introduced by Girard [5] in the late eighties. A logic is linear if every hypothesis is used exactly once. Thus, no hypothesis can be duplicated or removed at will, that is, the structural rules for contraction (duplication) and weakening (removal) must be banned from the logic. This property makes linear logic particularly suited for reasoning about state-based systems.

We enforce this property by restricting the inference rules in two ways. First, consider the identity axioms for OLL:

$$\frac{}{\cdot; A \vdash A} \text{id}_{\odot} \qquad \frac{}{A; \cdot \vdash A} \text{id}_{\triangleright}$$

The contexts are restricted to containing at most the hypothesis the axiom is proving holds. The other restriction is when applying the inference rules during the construction of derivations one is not allowed to copy or introduce new hypothesis across premises. This is why we take great care at separating the contexts in the definition of the inference rules.

Modeling structures in logic corresponds to interpreting the structures as formulas, and then reasoning about the structures corresponds to proving implications between the interpretations. As we will see, attack trees will be modeled as formulas of linear logic, and then we will prove several properties of attack trees by proving implications between them, but before moving on to ATLL I

give an example showing how to model a simple state-based system in linear logic.

There is a common example of problem solving used in artificial intelligence called the block world scenario; I learned of this example from Power and Webster's work on embedding linear logic in Coq [?]. Suppose there is a finite set of blocks and a robot arm that can be used to move blocks around. For example, the arm might break a single stack of blocks into two stacks. This setup can be modeled in linear logic using a few predicates on blocks.

The state of the system can be modeled using the following predicates:

$\text{on } x \ y$  : block  $x$  is on top of block  $y$   
 $\text{table } x$  : block  $x$  is on the table (i.e. no block underneath of  $x$ )  
 $\text{clear } x$  : there is no block on top of  $x$   
 $\text{holds } x$  : the robot arm is holding block  $x$   
 $\text{empty}$  : an atomic formula indicating the robot arm is not holding a block.

Next we must characterize the pre- and post-states of the valid actions one can perform during the game. These are defined as a set of linear logic axioms, where the hypothesis are the pre-state and the conclusion is the post-state. The following set of inference rules axiomatize the actions:

$$\frac{}{\vdash; \text{empty}, \text{clear } x \vdash (\text{holds } x) \odot (((\text{table } x) \multimap \top) \times ((\text{on } x \ y) \multimap (\text{clear } y)))} \text{get}$$

$$\frac{}{\vdash; \text{holds } x \vdash \text{empty} \odot (\text{clear } x) \odot ((\text{table } x) \times ((\text{clear } y) \multimap (\text{on } x \ y)))} \text{put}$$

The *get* axiom states that if the robot arm was empty and there was no block on top of  $x$ , then after the action the robot arm will be holding the block  $x$ , and either the block  $x$  had been on the table or it was on another block  $y$  which we now declare has no block on top of it. The *put* axiom states that if the robot arm was holding the block  $x$ , then after the action the robot arm is empty, the block  $x$  has no block on top of it, and either the block  $x$  is sitting on the table or it is sitting on the block  $y$  which was clear, and now we declare that  $x$  is on top of  $y$ .

We can see that the definition of the two axioms use the tensor product to indicate when two processes run in parallel. For example, in *get* the robot arm is holding the block  $x$  in parallel to checking the configuration of the table and other blocks. Then we use non-linear conjunction to model choices. Furthermore, the identity to the tensor product can be used to model processes that may consume a resource, but does not produce any new resources. For example, the formula  $(\text{table } x) \multimap \top$  consumes the resource corresponding to the block  $x$  being on the table, but then does not update the state.

At this point we can use our logic to prove properties about the block world scenario, and we can even derive new inference rules. For example, we could derive a new inference rule that swaps the two blocks on top of a stack of at least two, or that performs a get when there is a block underneath the block the

arm will pick up. We will use these insights throughout the remainder of this paper to model attack trees.

### 3 Attack Trees without Costs

In this section I introduce attack trees with sequential conjunction – sometimes referred to as SAND attack trees – without cost annotations, and then the Attack Tree Linear Logic (ATLL). This formulation of attack trees was first proposed by Jhawar et al. [6].

**Definition 1.** Suppose  $\mathbf{B}$  is a set of base attacks whose elements are denoted by  $b$ . Then an **attack tree** is defined by the following grammar:

$$T ::= b \mid T_1 \odot T_2 \mid T_1 \sqcup T_2 \mid T_1 \triangleright T_2$$

I denote unsynchronized parallel conjunction of attacks by  $T_1 \odot T_2$ , choice between attacks by  $T_1 \sqcup T_2$ , and sequential conjunction of attacks by  $T_1 \triangleright T_2$ .

Next I define ATLL and discuss how it can be used for reasoning about attack trees.

In ATLL attack trees are modeled as formulas, and thus, reasoning about attack trees should correspond to proving implications between them. ATLL is an extension of OLL (Section 2) with a new operator for modeling choice between attack trees. The ATLL syntax is defined by the following definition.

**Definition 2.** Suppose  $\mathbf{B}$  is a set of base attacks whose elements are denoted by  $b$ . The syntax of ATLL formulas and contexts are defined as follows:

$$\begin{aligned} \text{(formulas)} \quad E &::= b \mid E_1 \odot E_1 \mid E_1 \sqcup E_2 \mid E_1 \triangleright E_2 \mid E_1 \multimap E_2 \\ \text{(base contexts)} \quad \Gamma, \Delta &::= \cdot \mid b \mid \Gamma, \Delta \\ \text{(general contexts)} \quad \Theta, \Phi, \Psi &::= \cdot \mid E \mid \Gamma, \Delta \end{aligned}$$

$$\boxed{\begin{array}{c} \frac{}{b \vdash^T b} \text{id}_\odot \qquad \frac{\Gamma \vdash^T T_1 \quad \Delta \vdash^T T_2}{\Gamma, \Delta \vdash^T T_1 \odot T_2} \odot \qquad \frac{\Gamma \vdash^T T_1 \quad \Delta \vdash^T T_2}{\Gamma, \Delta \vdash^T T_1 \triangleright T_2} \triangleright \\ \frac{\Gamma \vdash^T T_1 \quad \Delta \vdash^T T_2}{\Gamma, \Delta \vdash^T T_1 \sqcup T_2} \sqcup \end{array}}$$

**Fig. 3.** Well Formed Attack Trees in ATLL

One can also view ATLL formulas as being defined as an extension of the syntax for attack trees with linear implication denoted by  $A \multimap B$ . Suppose  $E$  is an ATLL formula such that every base attack in  $E$  is in the base context  $\Gamma$ .



Then  $E$  is an attack tree if and only if  $\Gamma \vdash^T E$  holds with respect to the rules in Figure 3. These inference rules allow one to restrict themselves to attack trees when using ATLL.

The inference rules for in Figure 3 enforce several properties. First, they enforce that every base attack is used exactly once, and that an attack tree does not mention linear implication. Furthermore, they allow us to restrict our reasoning to attack trees in ATLL which I enforce in the hope that it will result in our reasoning being more amenable to automation, but this is left to future work.

$$\begin{array}{c}
\frac{}{\cdot; \cdot; E \vdash E} \text{id}_\odot \qquad \frac{}{\cdot; E; \cdot \vdash E} \text{id}_\sqcup \qquad \frac{}{E; \cdot; \cdot \vdash E} \text{id}_\triangleright \\
\\
\frac{\Theta_1; \Phi_1; \Psi_1 \vdash E_1 \quad \Theta_2; \Phi_2; \Psi_2 \vdash E_2}{\Theta_1, \Theta_2; \Phi_1, \Phi_2; \Psi_1, \Psi_2 \vdash E_1 \odot E_2} \odot_I \\
\\
\frac{\Theta_1; \Phi_1; \Psi_2 \vdash E_1 \odot E_2 \quad \Theta_2; \Phi_2; \Psi_1, E_1, E_2, \Psi_3 \vdash E_3}{\Theta_1, \Theta_2; \Phi_1, \Phi_2; \Psi_1, \Psi_2, \Psi_3 \vdash E_3} \odot_E \\
\\
\frac{\Theta_1; \Phi_1; \Psi_1 \vdash E_1 \quad \Theta_2; \Phi_2; \Psi_2 \vdash E_2}{\Theta_1, \Theta_2; \Phi_1, \Phi_2; \Psi_1, \Psi_2 \vdash E_1 \sqcup E_2} \sqcup_I \\
\\
\frac{\Theta_1; \Phi_2; \Psi_1 \vdash E_1 \sqcup E_2 \quad \Theta_2; \Phi_1, E_1, E_2, \Phi_3; \Psi_2 \vdash E_3}{\Theta_1, \Theta_2; \Phi_1, \Phi_2, \Phi_3; \Psi_1, \Psi_2 \vdash E_3} \sqcup_E \\
\\
\frac{\Theta_1; \Phi_1; \Psi_1 \vdash E_1 \quad \Theta_2; \Phi_2; \Psi_2 \vdash E_2}{\Theta_1, \Theta_2; \Phi_1, \Phi_2; \Psi_1, \Psi_2 \vdash E_1 \triangleright E_2} \triangleright_I \\
\\
\frac{\Theta_2; \Phi_1; \Psi_1 \vdash E_1 \triangleright E_2 \quad \Theta_1, E_1, E_2, \Theta_3; \Phi_2; \Psi_2 \vdash E_3}{\Theta_1, \Theta_2, \Theta_3; \Phi_1, \Phi_2; \Psi_1, \Psi_2 \vdash E_3} \triangleright_E \qquad \frac{\Theta; \Phi; \Psi_1, E_1, E_2, \Psi_2 \vdash E}{\Theta; \Phi; \Psi_1, E_2, E_1, \Psi_2 \vdash E} \text{sym}_\odot \\
\\
\frac{\Theta_1; \Phi_1; \Psi_1 \vdash E_1 \quad \Theta_2; \Phi_2; \Psi_2 \vdash E_2}{\Theta_1, \Theta_2; \Phi_1, \Phi_2; \Psi_1, \Psi_2 \vdash E_1 \sqcup E_2} \sqcup_I \\
\\
\frac{\Theta_1; \Phi_2; \Psi_1 \vdash E_1 \sqcup E_2 \quad \Theta_2; \Phi_1, E_1, E_2, \Phi_3; \Psi_2 \vdash E_3}{\Theta_1, \Theta_2; \Phi_1, \Phi_2, \Phi_3; \Psi_1, \Psi_2 \vdash E_3} \sqcup_E \qquad \frac{\Theta; \Phi_1, E_1, E_2, \Phi_2; \Psi \vdash E}{\Theta; \Phi_1, E_2, E_1, \Phi_2; \Psi \vdash E} \text{sym}_\sqcup \\
\\
\frac{\Theta; \Phi_1, (E_1, c_1), \Phi_2; \Psi \vdash_{c_2} E_2}{\Theta; \Phi_1, (E_1, c_1), (E_1, c_1), \Phi_2; \Psi \vdash_{c_2} E_2} \text{dup}_\sqcup \qquad \frac{\Theta; \Phi_1, (E_1, c_1), (E_1, c_1), \Phi_2; \Psi \vdash_{c_2} E_2}{\Theta; \Phi_1, (E_1, c_1), \Phi_2; \Psi \vdash_{c_2} E_2} \text{cont}_\sqcup \\
\\
\frac{\Theta; \Phi; \Psi, E_1 \vdash E_2}{\Theta; \Phi; \Psi \vdash E_1 \multimap E_2} \multimap_I \qquad \frac{\Theta_1; \Phi_1; \Psi_1 \vdash E_1 \multimap E_2 \quad \Theta_2; \Phi_2; \Psi_2 \vdash E_1}{\Theta_1, \Theta_2; \Phi_1, \Phi_2; \Psi_1, \Psi_2 \vdash E_2} \multimap_E \\
\\
\frac{\Theta_2; \Phi_1; \Psi_1 \vdash E_1 \multimap E_2 \quad \Theta_1; \Phi_2; \Psi_2 \vdash E_2 \multimap E_3}{\Theta_1, \Theta_2; \Phi_1, \Phi_2; \Psi_1, \Psi_2 \vdash E_1 \multimap E_3} \text{comp}_{\multimap}
\end{array}$$

**Fig. 4.** ATLL Logical Inference Rules

The ATLL logical inference rules are defined in Figure 4. The sequent of this system is denoted by  $\Theta; \Phi; \Psi \vdash E$  and consists of the general context  $\Theta$  associated with sequential conjunction,  $\Phi$  associated with choice, and  $\Psi$  associated with

$$\begin{array}{c}
\frac{\Theta, \Phi, \Psi \vdash^T T_1 \odot (T_2 \triangleright T_3)}{\Theta; \Phi; \Psi \vdash (T_1 \odot (T_2 \sqcup T_3)) \multimap ((T_1 \odot T_2) \sqcup (T_1 \odot T_3))} \text{dis}_{\odot_1} \\
\\
\frac{\Theta \vdash^T (T_2 \triangleright T_3) \odot T_1}{\Theta; \Phi; \Psi \vdash ((T_2 \sqcup T_3) \odot T_1) \multimap ((T_2 \odot T_1) \sqcup (T_3 \odot T_1))} \text{dis}_{\odot_2} \\
\\
\frac{\Theta, \Phi, \Psi \vdash^T T_1 \triangleright (T_2 \sqcup T_3)}{\Theta; \Phi; \Psi \vdash (T_1 \triangleright (T_2 \sqcup T_3)) \multimap ((T_1 \triangleright T_2) \sqcup (T_1 \triangleright T_3))} \text{dis}_{\triangleright_1} \\
\\
\frac{\Theta, \Phi, \Psi \vdash^T (T_2 \sqcup T_3) \triangleright T_1}{\Theta; \Phi; \Psi \vdash ((T_2 \sqcup T_3) \triangleright T_1) \multimap ((T_2 \triangleright T_1) \sqcup (T_3 \triangleright T_1))} \text{dis}_{\triangleright_2}
\end{array}$$

**Fig. 5.** ATLL Attack Tree Axioms

parallel conjunction. This is similar to OLL; see Section 2. It is required for hypotheses used with choice to be separated, because the attack tree equations for choice require contraction and a weak form of weakening; see the rules  $\text{dup}_{\sqcup}$  and  $\text{cont}_{\sqcup}$ .

These rules make up a basic linear logic for attack trees. The reader should note that the rule  $\text{comp}_{\multimap}$  is very useful when proving properties of attack trees, but is actually a derivable rule. Furthermore, one can see quite clearly that these rules are an extension of the rules for OLL, but instead of non-linear conjunction it has non-linear choice. In ATLL one can prove that choice, sequential conjunction, and parallel conjunction are associative, and choice and parallel conjunction are symmetric. However, one cannot prove the usual distributive laws associated with attack trees with sequential conjunction [6].

To overcome this limitation we add the attack tree axioms in Figure 5 to ATLL. The logical connective  $\multimap$  is linear biimplication which is defined in the usual way as classical implication. The reader should note that these axioms are restricted to attack trees only and not applicable to full ATLL formulas. In addition, the rule  $\text{dis}_{\odot_2}$  is derivable, because parallel conjunction is symmetric.

ATLL allows one to prove every equivalence for attack trees as linear implications that one might expect, in fact, the following are all provable in ATLL:

$$\begin{array}{ll}
T \sqcup T \multimap T & T_1 \odot (T_2 \sqcup T_3) \multimap (T_1 \odot T_2) \sqcup (T_1 \odot T_3) \\
T_1 \odot T_2 \multimap T_2 \odot T_1 & (T_2 \sqcup T_3) \odot T_1 \multimap (T_2 \odot T_1) \sqcup (T_3 \odot T_1) \\
T_1 \sqcup T_2 \multimap T_2 \sqcup T_1 & T_1 \triangleright (T_2 \sqcup T_3) \multimap (T_1 \triangleright T_2) \sqcup (T_1 \triangleright T_3) \\
(T_1 \sqcup T_2) \sqcup T_3 \multimap T_1 \sqcup (T_2 \sqcup T_3) & (T_2 \sqcup T_3) \triangleright T_1 \multimap (T_2 \triangleright T_1) \sqcup (T_3 \triangleright T_1) \\
(T_1 \odot T_2) \odot T_3 \multimap T_1 \odot (T_2 \odot T_3) & (T_1 \triangleright T_2) \triangleright T_3 \multimap T_1 \triangleright (T_2 \triangleright T_3)
\end{array}$$

These are all of the equations given by Jhawar et al. [6]. I conjecture that it should be possible to define a proof search algorithm for ATLL to automate equational reasoning on attack trees, but this is future work. Using implication as a means to reason about attack trees opens the door for different types of reasoning.

Suppose  $T_1$  is a larger attack tree, and we wish to know if the attack tree  $T_2$  is contained within  $T_1$ , that is,  $T_2$  is a subattack tree of  $T_1$ . Currently, ATLL will not allow us to prove this, but it can if we add the following rules:

$$\boxed{\begin{array}{c} \frac{\Theta_1, \Theta_2; \Phi; \Psi \vdash E \quad \Delta \vdash^T T}{\Theta_1, T, \Theta_2; \Phi; \Psi \vdash E} \text{wk}_{\triangleright} \qquad \frac{\Theta; \Phi_1, \Phi_2; \Psi \vdash E \quad \Delta \vdash^T T}{\Theta; \Phi_1, T, \Phi_2; \Psi \vdash E} \text{wk}_{\sqcup} \\[10pt] \frac{\Theta; \Phi; \Psi_1, \Psi_2 \vdash E \quad \Delta \vdash^T T}{\Theta; \Phi; \Psi_1, T, \Psi_2 \vdash E} \text{wk}_{\odot} \end{array}}$$

These rules allow us to discard parts of a larger attack tree while constructing proof derivations, but this only works for attack trees not general ATLL formulas. Thus, with these rules ATLL becomes slightly affine. We can now use these rules to determine if  $T_1 \multimap T_2$  holds, and if it does, then  $T_2$  is a subattack tree of  $T_1$ .

I now give an example to illustrate using the system to prove properties of attack trees. Consider the two example trees from the introduction:

$$\begin{aligned} T_1 &:= (B_1 \odot (B_2 \sqcup B_3)) \triangleright B_4 \\ T_2 &:= ((B_1 \odot B_2) \triangleright B_4) \sqcup ((B_1 \odot B_3) \triangleright B_4) \end{aligned}$$

The attack trees  $T_1$  and  $T_2$  are equivalent if and only if  $T_1 \multimap T_2$  holds. First, ATLL attack tree axioms are bimplications, and hence, we may treat these axioms as equivalences. Thus, using the ATLL attack tree axioms we can prove that  $T_1$  is equivalent to  $T_2$  informally as follows:

$$\begin{aligned} T_1 &:= (B_1 \odot (B_2 \sqcup B_3)) \triangleright B_4 && \text{(definition)} \\ &\multimap ((B_1 \odot B_2) \sqcup (B_1 \odot B_3)) \triangleright B_4 && \text{(rule dis}_{\odot_1}) \\ &\multimap ((B_1 \odot B_2) \triangleright B_4) \sqcup ((B_1 \odot B_3) \triangleright B_4) && \text{(rule dis}_{\triangleright_2}) \\ &:= T_2 && \text{(definition)} \end{aligned}$$

The corresponding formal derivation is quite large, but is easily constructible using the previous reasoning.

## 4 Attack Trees with Costs

Now I generalize Bucciarelli and Ehrhards indexed linear logic [2] to obtain an extension of ATLL called Indexed ATLL with the ability to model and reason about attack trees with costs.

**Definition 3.** Suppose  $\mathbf{B}$  is a set of base attacks whose elements are denoted by  $b$ , and  $\mathbf{C}$  is a set of costs whose elements are denoted by  $c$ . Additionally, let  $(\mathbf{C}, \text{op}_{\odot})$  and  $(\mathbf{C}, \text{op}_{\sqcup})$  be symmetric monoids on  $\mathbf{C}$ , and  $(\mathbf{C}, \text{op}_{\triangleright})$  be a monoid on  $\mathbf{C}$ , where  $\text{op}_{\odot}$  and  $\text{op}_{\triangleright}$  distribute over  $\text{op}_{\sqcup}$ . Then an **attack tree with costs** is defined by the following grammar:

$$T ::= (b, c) \mid T_1 \odot T_2 \mid T_1 \sqcup T_2 \mid T_1 \triangleright T_2$$

$$\begin{array}{c}
\frac{}{(b, c) \vdash_c^T b} \text{id}_\odot \qquad \frac{\Delta_1 \vdash_{c_1}^T T_1 \quad \Delta_2 \vdash_{c_2}^T T_2}{\Delta_1, \Delta_2 \vdash_{\text{op}_\odot(c_1, c_2)}^T T_1 \odot T_2} \odot \\
\\
\frac{\Delta_1 \vdash_{c_1}^T T_1 \quad \Delta_2 \vdash_{c_2}^T T_2}{\Delta_1, \Delta_2 \vdash_{\text{op}_\triangleright(c_1, c_2)}^T T_1 \triangleright T_2} \triangleright \qquad \frac{\Delta_1 \vdash_{c_1}^T T_1 \quad \Delta_2 \vdash_{c_2}^T T_2}{\Delta_1, \Delta_2 \vdash_{\text{op}_\sqcup(c_1, c_2)}^T T_1 \sqcup T_2} \sqcup
\end{array}$$

**Fig. 6.** Well Formed Attack Trees in Indexed ATLL

Each base attack is annotated with a cost, and every branching node is associated with a binary operation for computing the cost at that node using the costs of the left subtree and the right subtree.

I now extend the syntax and rules of ATLL into Indexed ATLL, but due to space I do not give every rule here, but only the most interesting ones, because they are similar to ATLL; the entire system is defined in Appendix A.

**Definition 4.** Suppose  $\mathcal{B}$  is a set of base attacks whose elements are denoted by  $b$ , and  $\mathcal{C}$  is a set of costs whose elements are denoted by  $c$ . Additionally, let  $(\mathcal{C}, \text{op}_\odot)$  and  $(\mathcal{C}, \text{op}_\sqcup)$  be symmetric monoids on  $\mathcal{C}$ , and  $(\mathcal{C}, \text{op}_\triangleright)$  be a monoid on  $\mathcal{C}$ , where  $\text{op}_\odot$  and  $\text{op}_\triangleright$  distribute over  $\text{op}_\sqcup$ . Furthermore, let  $(\mathcal{C}, \text{rel}_\circ)$  be a preorder on  $\mathcal{C}$ . Then the syntax of Indexed ATLL formulas and contexts are defined as follows:

$$\begin{aligned}
(\text{formulas}) \quad E &::= (b, c) \mid E_1 \odot E_2 \mid E_1 \sqcup E_2 \mid E_1 \triangleright E_2 \mid (E_1, c) \multimap E_2 \\
(\text{base contexts}) \quad \Gamma, \Delta &::= \cdot \mid (b, c) \mid \Gamma, \Delta \\
(\text{general contexts}) \quad \Theta, \Psi &::= \cdot \mid (E, c) \mid \Gamma, \Delta
\end{aligned}$$

As we can see the only major changes to the ATLL syntax is that all hypotheses are annotated with costs.

The inference rules for well-formed attack trees in Indexed ATLL are given in Figure 6. The sequent is now denoted by  $\Delta \vdash_c^T T$  where  $c$  is the fully computed cost of the root of the attack tree  $T$ . Notice that this cost is computed during the construction of a proof derivation. This cost will be used along with the relation on implication to influence the relationship between attack trees.

The logical inference rules for Indexed ATLL are given in Figure 7. As I mentioned above I only give a select few of the rules here, but the complete system can be found in Appendix A. The most interesting rule is the introduction rule for implication,  $\multimap_I$ , because the second premise is  $\text{rel}_\circ(c_1, c_2)$ .

Suppose  $(T_1, c_1)$  and  $(T_2, c_2)$  are two attack trees with the respective costs, and the relation  $\text{rel}_\circ(c_1, c_2)$  holds if and only if  $c_1 \leq c_2$ . If we wanted to show that the attack tree  $T_1$  is equivalent to the attack tree  $T_2$ , then we would have to show  $T_1 \multimap T_2$  where  $\text{rel}_\circ(c_1, c_2)$  holds if and only if  $\text{rel}_\circ(c_1, c_2)$  and  $\text{rel}_\circ(c_2, c_1)$  hold. This would require one to prove  $\text{rel}_\circ(c_1, c_2)$  which would require  $c_1 = c_2$ . Thus, in this example, attack trees would be equivalent if and

$$\begin{array}{c}
\frac{}{\cdot; \cdot; (E, c) \vdash_c E} \text{id}_\odot \qquad \frac{}{\cdot; (E, c); \cdot \vdash_c E} \text{id}_\sqcup \qquad \frac{}{(E, c); \cdot; \cdot \vdash_c E} \text{id}_\triangleright \\
\\
\frac{\Theta_1; \Phi_1; \Psi_1 \vdash_{c_1} E_1 \quad \Theta_2; \Phi_2; \Psi_2 \vdash_{c_2} E_2}{\Theta_1, \Theta_2; \Phi_1, \Phi_2; \Psi_1, \Psi_2 \vdash_{\text{op}_\triangleright(c_1, c_2)} E_1 \triangleright E_2} \triangleright_I \\
\\
\frac{\Theta_2; \Phi_1; \Psi_2 \vdash_{\text{op}_\triangleright(c_1, c_2)} E_1 \triangleright E_2 \quad \Theta_1, (E_1, c_1), (E_2, c_2), \Theta_3; \Phi_2; \Psi_2 \vdash_{c_3} E_3}{\Theta_1, \Theta_2, \Theta_3; \Phi_1, \Phi_2; \Psi_1, \Psi_2 \vdash_{c_3} E_3} \triangleright_E \\
\\
\frac{\Theta; \Phi; \Psi, (E_1, c_1) \vdash_{c_2} E_2 \quad \text{rel}_\multimap(c_1, c_2)}{\Theta; \Phi; \Psi \vdash_{c_2} (E_1, c_1) \multimap E_2} \multimap_I \\
\\
\frac{\Theta_1; \Phi_1; \Psi_1 \vdash_{c_2} (E_1, c_1) \multimap E_2 \quad \Theta_2; \Phi_2; \Psi_2 \vdash_{c_1} E_1}{\Theta_1, \Theta_2; \Phi_1, \Phi_2; \Psi_1, \Psi_2 \vdash_{c_2} E_2} \multimap_E
\end{array}$$

**Fig. 7.** Select Logical Inference Rules for Indexed ATLL

only if they have the same logical structure and the same overall cost. Thus, several different types of reasoning on attack trees can be modeled in Index ATLL depending on the relation chosen for implication.

## 5 Related Work

TODO

## 6 Conclusion and Future Work

I introduced a new linear logic for reasoning about attack trees called the Attack Tree Linear Logic (ATLL) with and without costs. Attack trees are modeled as logical formulas and proving relationships between attack trees corresponds to proving implications. Adding costs to ATLL results in a new indexed linear logic called Indexed ATLL. Then I showed how endowing implication with a relation on costs can be used to include the costs when reasoning about attack trees.

ATLL is just one part of this project, the following are some next steps with respect to (Indexed) ATLL and other parts of the project:

- Define a term assignment for (Indexed) ATLL that can be used to define attack trees using a functional programming language.
- Determine the applicability of automating reasoning on attack trees in (Indexed) ATLL using proof search.
- Finish studying the categorical model to which (Indexed) ATLL is based.
- Finishing the development of a modular and stand alone IDE for attack tree in Javascript that can be used with various formalizations of attack trees. Hopefully, the community will be able to make use of this interface.
- Develop the concrete implementation, called Lina, of (Indexed) ATLL with automation.

## References

1. Carolyn Brown, Doug Gurr, and Valeria Paiva. A linear specification language for petri nets. *DAIMI Report Series*, 20(363), 1991.
2. Antonio Bucciarelli and Thomas Ehrhard. On phase semantics and denotational semantics in multiplicative-additive linear logic. *Annals of Pure and Applied Logic*, 102(3):247 – 282, 2000.
3. Marcelo Fiore and Marco Devesas Campos. *Computation, Logic, Games, and Quantum Foundations. The Many Facets of Samson Abramsky: Essays Dedicated to Samson Abramsky on the Occasion of His 60th Birthday*, chapter The Algebra of Directed Acyclic Graphs, pages 37–51. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
4. Luisa Francesco Albasini, Nicoletta Sabadini, and Robert F. C. Walters. The compositional construction of markov processes. *Applied Categorical Structures*, 19(1):425–437, 2010.
5. Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1 – 101, 1987.
6. Ravi Jhavar, Barbara Kordy, Sjouke Mauw, Saša Radomirović, and Rolando Trujillo-Rasua. Attack trees with sequential conjunction. In Hannes Federrath and Dieter Gollmann, editors, *ICT Systems Security and Privacy Protection*, volume 455 of *IFIP Advances in Information and Communication Technology*, pages 339–353. Springer International Publishing, 2015.
7. Barbara Kordy, Marc Pouly, and Patrick Schweitzer. Computational aspects of attack-defense trees. In Pascal Bouvry, Mirosław Kłopotek, Franck Leprévost, Małgorzata Marciniak, Agnieszka Mykowiecka, and Henryk Rybiński, editors, *Security and Intelligent Information Systems*, volume 7053 of *Lecture Notes in Computer Science*, pages 103–116. Springer Berlin Heidelberg, 2012.
8. Paul-André Melliès. Categorical semantics of linear logic. In Pierre-Louis Curien, Hugo Herbelin, Jean-Louis Krivine, and Paul-André Melliès, editors, *Interactive models of computation and program behaviour*. Panoramas et Synthèses 27, Société Mathématique de France, 2009.
9. Jeff Polakow. *Ordered linear logic and applications*. PhD thesis, Carnegie Mellon University, 2001.
10. A Tzouvaras. The linear logic of multisets. *Logic Journal of IGPL*, 6(6):901–916, 1998.

## Appendix

### A The Full Specification of Indexed ATLL

**Definition 4.** Suppose  $B$  is a set of base attacks whose elements are denoted by  $b$ , and  $C$  is a set of costs whose elements are denoted by  $c$ . Additionally, let  $(C, \text{op}_\odot)$  and  $(C, \text{op}_\sqcup)$  be symmetric monoids on  $C$ , and  $(C, \text{op}_\triangleright)$  be a monoid on  $C$ , where  $\text{op}_\odot$  and  $\text{op}_\triangleright$  distribute over  $\text{op}_\sqcup$ . Furthermore, let  $(C, \text{rel}_\multimap)$  be a preorder on  $C$ . Then the syntax of Indexed ATLL formulas and contexts are defined as follows:

$$\begin{aligned}
 (\text{formulas}) \quad E &::= (b, c) \mid E_1 \odot E_2 \mid E_1 \sqcup E_2 \mid E_1 \triangleright E_2 \mid (E_1, c) \multimap E_2 \\
 (\text{base contexts}) \quad \Gamma, \Delta &::= \cdot \mid (b, c) \mid \Gamma, \Delta \\
 (\text{general contexts}) \quad \Theta, \Psi &::= \cdot \mid (E, c) \mid \Gamma, \Delta
 \end{aligned}$$

$\frac{}{(b, c) \vdash_c^T b} \text{id}_\odot$	$\frac{\Delta_1 \vdash_{c_1}^T T_1 \quad \Delta_2 \vdash_{c_2}^T T_2}{\Delta_1, \Delta_2 \vdash_{\text{op}_\odot(c_1, c_2)}^T T_1 \odot T_2} \odot$
$\frac{\Delta_1 \vdash_{c_1}^T T_1 \quad \Delta_2 \vdash_{c_2}^T T_2}{\Delta_1, \Delta_2 \vdash_{\text{op}_\triangleright(c_1, c_2)}^T T_1 \triangleright T_2} \triangleright$	$\frac{\Delta_1 \vdash_{c_1}^T T_1 \quad \Delta_2 \vdash_{c_2}^T T_2}{\Delta_1, \Delta_2 \vdash_{\text{op}_\sqcup(c_1, c_2)}^T T_1 \sqcup T_2} \sqcup$

**Table 1.** Well Formed Attack Trees in Indexed ATLL

$\frac{}{\cdot; \cdot; (E, c) \vdash_c E} \text{id}_\odot$	$\frac{}{\cdot; (E, c); \cdot \vdash_c E} \text{id}_\sqcup$	$\frac{}{(E, c); \cdot; \cdot \vdash_c E} \text{id}_\triangleright$
$\frac{\Theta_1; \Phi_1; \Psi_1 \vdash_{c_1} E_1 \quad \Theta_2; \Phi_2; \Psi_2 \vdash_{c_2} E_2}{\Theta_1, \Theta_2; \Phi_1, \Phi_2; \Psi_1, \Psi_2 \vdash_{\text{op}_\odot(c_1, c_2)} E_1 \odot E_2} \odot_I$		
$\frac{\Theta_1; \Phi_1; \Psi_2 \vdash_{\text{op}_\odot(c_1, c_2)} E_1 \odot E_2 \quad \Theta_2; \Phi_2; \Psi_1, (E_1, c_1), (E_2, c_2), \Psi_3 \vdash_{c_3} E_3}{\Theta_1, \Theta_2; \Phi_1, \Phi_2; \Psi_1, \Psi_2, \Psi_3 \vdash_{c_3} E_3} \odot_E$		
$\frac{\Theta_1; \Phi_1; \Psi_1 \vdash_{c_1} E_1 \quad \Theta_2; \Phi_2; \Psi_2 \vdash_{c_2} E_2}{\Theta_1, \Theta_2; \Phi_1, \Phi_2; \Psi_1, \Psi_2 \vdash_{\text{op}_\sqcup(c_1, c_2)} E_1 \sqcup E_2} E\_choiI$		
$\frac{\Theta_1; \Phi_2; \Psi_1 \vdash_{\text{op}_\sqcup(c_1, c_2)} E_1 \sqcup E_2 \quad \Theta_2; \Phi_1, (E_1, c_1), (E_2, c_2), \Phi_3; \Psi_2 \vdash_{c_3} E_3}{\Theta_1, \Theta_2; \Phi_1, \Phi_2, \Phi_3; \Psi_1, \Psi_2 \vdash_{c_3} E_3} E\_choiE$		
$\frac{\Theta_1; \Phi_1; \Psi_1 \vdash_{c_1} E_1 \quad \Theta_2; \Phi_2; \Psi_2 \vdash_{c_2} E_2}{\Theta_1, \Theta_2; \Phi_1, \Phi_2; \Psi_1, \Psi_2 \vdash_{\text{op}_\triangleright(c_1, c_2)} E_1 \triangleright E_2} \triangleright_I$		
$\frac{\Theta_2; \Phi_1; \Psi_2 \vdash_{\text{op}_\triangleright(c_1, c_2)} E_1 \triangleright E_2 \quad \Theta_1, (E_1, c_1), (E_2, c_2), \Theta_3; \Phi_2; \Psi_2 \vdash_{c_3} E_3}{\Theta_1, \Theta_2, \Theta_3; \Phi_1, \Phi_2; \Psi_1, \Psi_2 \vdash_{c_3} E_3} \triangleright_E$		
$\frac{\Theta; \Phi; \Psi_1, (E_1, c_1), (E_2, c_2), \Psi_2 \vdash_c E}{\Theta; \Phi; \Psi_1, (E_2, c_2), (E_1, c_1), \Psi_2 \vdash_c E} \text{sym}_\odot$		
$\frac{\Theta; \Phi_1, (E_1, c_1), (E_2, c_2), \Phi_2; \Psi \vdash_c E}{\Theta; \Phi_1, (E_2, c_2), (E_1, c_1), \Phi_2; \Psi \vdash_c E} \text{sym}_\sqcup$		
$\frac{\Theta; \Phi_1, (E_1, c_1), \Phi_2; \Psi \vdash_{c_2} E_2}{\Theta; \Phi_1, (E_1, c_1), (E_1, c_1), \Phi_2; \Psi \vdash_{c_2} E_2} \text{dup}_\sqcup$		
$\frac{\Theta; \Phi_1, (E_1, c_1), (E_1, c_1), \Phi_2; \Psi \vdash_{c_2} E_2}{\Theta; \Phi_1, (E_1, c_1), \Phi_2; \Psi \vdash_{c_2} E_2} \text{cont}_\sqcup$		
$\frac{\Theta; \Phi; \Psi, (E_1, c_1) \vdash_{c_2} E_2 \quad \text{rel}_{\multimap}(c_1, c_2)}{\Theta; \Phi; \Psi \vdash_{c_2} (E_1, c_1) \multimap E_2} \multimap_I$		
$\frac{\Theta_1; \Phi_1; \Psi_1 \vdash_{c_2} (E_1, c_1) \multimap E_2 \quad \Theta_2; \Phi_2; \Psi_2 \vdash_{c_1} E_1}{\Theta_1, \Theta_2; \Phi_1, \Phi_2; \Psi_1, \Psi_2 \vdash_{c_2} E_2} \multimap_E$		

**Table 2.** Logical Inference Rules for Indexed ATLL



$$\begin{array}{c}
\frac{\Theta, \Phi, \Psi \vdash_c^T T_1 \odot (T_2 \triangleright T_3)}{\Theta; \Phi; \Psi \vdash_c (T_1 \odot (T_2 \sqcup T_3)) \multimap ((T_1 \odot T_2) \sqcup (T_1 \odot T_3))} \text{dis}_{\odot_1} \\
\\
\frac{\Theta, \Phi, \Psi \vdash_c^T (T_2 \triangleright T_3) \odot T_1}{\Theta; \Phi; \Psi \vdash_c ((T_2 \sqcup T_3) \odot T_1) \multimap ((T_2 \odot T_1) \sqcup (T_3 \odot T_1))} \text{dis}_{\odot_2} \\
\\
\frac{\Theta, \Phi, \Psi \vdash_c^T T_1 \triangleright (T_2 \sqcup T_3)}{\Theta; \Phi; \Psi \vdash_c (T_1 \triangleright (T_2 \sqcup T_3)) \multimap ((T_1 \triangleright T_2) \sqcup (T_1 \triangleright T_3))} \text{dis}_{\triangleright_1} \\
\\
\frac{\Theta, \Phi, \Psi \vdash_c^T (T_2 \sqcup T_3) \triangleright T_1}{\Theta; \Phi; \Psi \vdash_c ((T_2 \sqcup T_3) \triangleright T_1) \multimap ((T_2 \triangleright T_1) \sqcup (T_3 \triangleright T_1))} \text{dis}_{\triangleright_2}
\end{array}$$

**Table 3.** Indexed ATLL Attack Tree Axioms